Diary:

**Day 4 (Monday 27/01):**

Created Repo without using CI template.

Added 2 project boards – one for the user stories and one for the learning objectives.

Used Richard’s walkthrough to get myself set up using VS Code on my desktop.

Worked through the “I think therefore I blog” project to get to the Heroku deployment stage before lunch.

**Copilot** assisted me with Heroku deployment issue. I had copied the text from the blog project into the Procfile file and it referred to “codestar” rather than “allcoopedup”.

Added skeletons for README and TESTING files.

Generated 5 reviews using **Copilot** to use for initial database entries.

I remembered about there being a section of the blog tutorial where dummy data was bulk loaded. Fixtures help you to do this. The page which discusses them is:

<https://learn.codeinstitute.net/courses/course-v1:CodeInstitute+FSD101_WTS+4/courseware/713441aba05441dfb3a7cf04f3268b3f/6dda9f5767854b97a9f9dd9d6efc7e01/?child=first>

The source code sheet to use as a template for the data is:

<https://github.com/Code-Institute-Solutions/blog/blob/599c708792d14a883a081e16539a1a45c9d1c997/07_Rich_text_reload/03_adding_more_posts/blog/fixtures/posts.json#L4>

If I struggle to create the JSON file, this might work: <https://tableconvert.com/excel-to-json>

**Day 5 (Tuesday 28/01):**

Had a few issues starting up this morning. I noticed that I wasn’t in my virtual environment. I thought I had entered into it by entering a command in powershell, but apparently not (none of my packages were appearing in “requirements.txt”). I manually reloaded the packages and then correctly restarted the venv, and when I pushed it to Github it said there were 7,511 files and many duplicates which is deleted (presumably the package files which were then in the venv twice).

I used **Copilot** to ask it about whether I need to start the venv for each session and to explain why that would need to be done, along with the right code to start it in different terminal types.

Once that all settled down, I started my database and added the secret key before stand up.

I asked Emma to have a look at my ERDs so that I can ensure that they’re right before I code my models.

Emma helped by highlighting that although I don’t need to physically code a User model (as it’s imported from Django) it needs to be in my ERDs. We tweaked some fields to correct them (putting the Author as a Foreign Key for example) and amending the PEGI and star ratings to be drop down fields. She also showed me the code for how to do this using her own repo and also introduced me to emojipedia so that I can use emojis for the stars in the star ratings.

A screenshot of a computer program

Description automatically generated

Creating my first model:

Seems straight forward enough, though when adding the code for the user it included a “related name” from the tutorial which was “blog\_posts”. I used **Copilot** to remind me what the related name meant and what it should be in this project. I’ve put in a holding code of “review” for the moment, though this may need to be changed.

Whilst coding there are many times when **Copilot** ghost prompts appear, which I check for accuracy and accept (or edit). This seems to be better than typing the code myself as I’m more likely to key in typing errors, such as when I spelled “IntegerField” as “IntergerField” (which prevented migrations).

Set up admin and loaded up the form to see how it looks when adding a review.

The drop down fields have the 1st option in them which I’m not sure is what I wanted. It’s ok for an MVP, but if these fields are optional then I would want them to be blank rather than filled in with default answers (like every game that doesn’t have a star-rating selected will get 1 star). I therefore need to fix that.

**FIXES TO BE DONE:**

Ensure default is blank rather than drop-down option 1.

Configure mandatory fields

Added Comments model and configured Admin without much issue.

The game title is not showing properly yet, but I think that gets addressed in due course. Fairly easy fix I’m sure though.

Plus on the comments form you essentially put the game title in twice, which isn’t going to work. Will need to check once the fields have been taken care of and then sort it. That will likely mean purging the database, but that’s ok.

**FIXES TO BE DONE:**

Make sure review title, etc. is flowing through properly.

Reconfigure Comments model to get rid of putting in the title twice.

Creating views:

In urls in the allcoopedup directory, I’ve deleted:

“from review.views import my\_review” from the imports, and

“path('review/', my\_review, name='review'),” from the urlpatterns.

Added directory into review directory to be: review/templates/review, and then created an HTML file in there with some sample code which I believe I’ve customised to match my model.

From the views.py in the review directory, I’ve deleted: from django.http import HttpResponse and def my\_review(request):

    return HttpResponse('This is my review page').

Added Summernote. All still working great.

Manually added more AI generated reviews. Thought about using a JSON file, but realised it would take longer to add them into there than it would to do it manually. I realise that if I have to keep adding them to initialised databases the JSON file would be amazing, though I could probably do it 5 times before the JSON file would pay back, so I’ll go manual.

Looked into the extra field in the comment model with **Copilot** and realised that I need to rebuild my database. Need to find the section of the blog tutorial that handles that.

That’s for tomorrow.

**Day 6 (Wednesday 29/01):**

Started the day well. Found the section on “debugging the database” which included information on how to add/remove/edit fields in models and sort the database so that it doesn’t cause an error. I wrote a process sheet for it so that I can refer to it when it comes to adding images to the reviews. Will also need a default image for the moment. Probably just something that says “Box art missing”.

I decided whilst half asleep in bed this morning that the background for the site should be a vignette-style graduation from black at the edges/corners into a dark grey in the centre. Could I code a dark/light mode toggle? Maybe!

Created base.html and index.html by customising blog tutorial templates.

I really didn't think it would work, but it did. Pretty much perfectly. Of course, it’s someone else’s template and it needs a lot of customisation, but it works.

Just thinking now how the CSS file is going to be super customised to the blog project and how I’m going to have to either scrap it completely or spend ages tailoring it (and deleting bits that I’m not using).

**Copilot usage of the day!!!**

I needed an image as a holding image for testing (and for the circumstances of an uploaded image being missing. So I gave it this prompt:

A close up of black text

Description automatically generated

It gave me this:

A person in a red hoodie

Description automatically generated

So I said,



And it basically made me:

A cartoon of a person with glasses and a beard

Description automatically generated

I had an “Internal Server Error” when I tried to open my deployed app on Heroku at the “building a blog post” section of the tutorial. I got **Copilot** to work out what the error was by feeding it the error message from the terminal, and it highlighted an indentation issue in my views file. I couldn’t immediately see it, so I got it to apply the correction in the code and it highlighted where it was.

I had an issue where my full review text wasn’t coming through on the detailed review page, but I saw the error and fixed it.

So, it’s been a few hours and I’ve been trying to get the about page done. It’s not gone smoothly as there have been many issues due to my final code including many things (like crispy forms and the collaborate form) which weren’t included at this stage. I tried to play fast and loose with the code, but it didn’t work. I think I’m nearly there, but I’ve had to remove this code from the about.html page to see if I can get it to load:

    <!-- <div class="row justify-content-center">

        <div class="col-12 col-md-6 my-5">

            <h2>Let's collaborate!</h2>

            <p>I believe in the power of shared knowledge and collective effort.

                Whether you have a project in mind, wish to co-author

                an article, or simply want to brainstorm some ideas,

                I'm excited to hear from you. Fill out the form

                and we can get the ball rolling!</p> -->

            <!-- add your form here. Your submit button should

                                have the classes of btn, btn-secondary -->

            <!-- <form method="POST">

                {{ collaborate\_form | crispy }}

                {% csrf\_token %}

                <button class="btn btn-secondary" type="submit">Submit</button>

            </form>

        </div>

    </div>

</div> -->

Eventually just replaced the html code with the original. Now it works.

Authentication section now done (issues around the slashes in “cp -r C:/Users/tacki/Documents/GitHub/AllCoopedUp/venv/Lib/site-packages/allauth/templates/\* ./templates/” but eventually worked it out.

Customised login/logout/signup pages too. On to “Views 3” tomorrow.

**Day 7 (Thursday 30/01):**

Wow, a week in. I may have an MVP of the site done by the end of the week if I keep up this slow, but steady pace. I realised just now how much progress was made yesterday. I was thinking about what I’ve done on the site this morning before I started and was pretty convinced that the AI image was done on Tuesday. That sort of means that yesterday feels like 2 days’ work.

Today has started well again. I’ve started on Views Part 3 and had a confidence wobble as I amended my admin. I felt confident because I added an “approved” header to the reviews table, but less confident when I tried to put any sort of header over the comments and got an error. It’s ok. MVP. I could undo the code easily and I’ve made a note to add it if I can, but I feel like that’s something I can do on another project which won’t matter as much if I accidentally screw it up.

There were a few things to understand and change about the code being added to the review html (to get the comments loaded), especially as I don’t have “created\_on” fields. I’ve deleted what I think needs to be deleted and we seem golden for now.

Just added the comments form to my review page (wow, there are some chunky bits just in the first section of this tutorial module!) and I had errors. I realised I’d skipped over a couple of bits of code (amazing how I keep doing that, what with my legendary attention to detail), but was able to resolve them without seeking further assistance.

Now added the collaboration form. These bits where I don’t have to change much code are good, but I still have to go through the code, reading and understanding it to make sure it works. There was a moment that it didn’t work, and I was quickly able to identify that there was a missing comma, though only through error messages, code colour indicators, and a source code to compare to. Without all of that, and coding straight out of my head… Not something I could do yet, but I think that’s the difference between a junior developer and a senior dev.

**Copilot** helped me with resolving an error in the POST wiring of the collaborate form. See the Errors folder in the project.

I’ve now added edit and delete functionality to the comment form. I had to be careful with the JavaScript as there was a reference to “body” in there which in my project is “comment” but I saw it and recoded it. Am quite proud that I’ve got it to this stage.

I’ve also been ticking off items on the project board. There are a few in the “in progress” column which could be ticked off if I didn’t include both user review and comment functionality to them. The comment stuff is fairly straightforward and is done following the tutorial, but the review requires a lot more work. They’ll probably be in there until COP tomorrow. That’s my target anyway. If I get those done, that’s effectively my MVP, with the only must-haves relating to visuals.

Now added Cloudinary and got missing boxart to show in the index page. Had a Woo Hoo moment, then a feeling of sadness that if all goes according to plan, nobody will ever see my missingboxart image. So, for now, here it is in its full glory:

A screenshot of a video game

Description automatically generated

Soon afterwards it was replaced by:

A screenshot of a video game

Description automatically generated

I got the images working on the review detail screen too.

A screenshot of a video game

Description automatically generated

Some work better than others though and I’ll have to do some CSS wizardry to get them to be positioned correctly. Maybe. If I have time.

A close-up of a red object

Description automatically generated

And, holy heck… I’m at the end of the tutorial. The rest is testing.

Need to **remember** to go back to the video at the end of “Where to Put Things” to go through the initial testing.

<https://www.youtube.com/watch?v=9F0BpXzsh3Y>

I’m using a Miro board to complement the development process and make notes. I’m better in the mornings if I have a clear idea of what I need to be cracking on with, so for tomorrow I’ve added instructions to guide me on what to tackle next. It feels like stepping away from the guide rails and if I’m honest I’m a bit nervous. I think there will be self-doubt, great frustration if/when things go wrong, fear of ruining what I’ve created and being unable to get it back. On the flip side, the greater the risk the greater the reward. It’s out of my comfort zone, so I think when I get the code right and working I’ll be very buoyant and proud of myself. I mean, I’m pretty proud when I manage to make things work while following a tutorial, so I’ll likely be buzzing if I manage to code alone.

**Day 8 (Friday 31/01):**

I had a look at the deployed site on mobile and tablet yesterday evening and a few things came up. You could look at these as “**bugs**” which need to be fixed at some point, though could be outside of the scope of the project as they don’t hinder basic function of the site.

**Bug 1:** The cards aren’t equally spaced due to the differing sizes of the images.

**Bug 2:** The images aren’t optimised for the banner view in the detailed review page.

**Bug 3:** You should be able to click on the image to take you from the index page to the detailed review page. It’s better UX.

**Bug 4:** There are gaps on the detailed review page (especially on mobile) below the banner. Needs to be investigated as that may be where the images are not loading because they don’t fit on mobile.

In addition, the tablet view isn’t showing correctly. It looks like the mobile view, so needs to be fixed fairly promptly with CSS, Bootstrap or media queries. I’d go Bootstrap as that’s already being used in the template.

I’ve also changed my mind about today. I’m going to start with the review page. I’m just putting it off as it feels like a big thing to do. It would tick off many user stories and get me closer to the MVP, so is the right choice. Be bold and confident. It doesn’t need a new model or admin, just a form, template, and everything else linked up. Take it slowly!

I’m starting with a prompt to **Copilot** to see how much I can use it to help me with this coding. My initial prompt is:

*Just to set the context of what I'm doing today... I'm making a video game review site using python, Django, html, css and JavaScript (plus Bootstrap). I've been following a tutorial from a course I'm taking which has got me to a certain point, but I now need to add my own pages and functionality so that my project is different to that created in the tutorial. The first stage of that is to create a page where users can submit their own reviews. A model exists for this in the review/models.py file (class Review), and I know that I'll have to load most of the fields from that into a form for the user to fill in. The user needs to be registered and logged in to submit their review, and similar code exists in the Comments section of review/review\_detail.html to facilitate that (as well as confirming that the review has been submitted). I know I also need to create JavaScript code to allow the user to edit and delete their review, and the view of their review needs to be similarly greyed out (like the comments in review/review\_detail.html) when they are pending administrator's approval. I'd really like your help and guidance to get this done. I need to fully understand everything that you're suggesting and for it to be broken down into multiple steps so that I can follow it. I'm very new to coding and all of the languages I've mentioned above, so please treat me as a beginner. Are you able to help me with this please?*

So, this is interesting. Left to my own devices I would have started with the “user\_review.html” page as I felt like that was the least destructive approach, though **Copilot** has gone a different (and way more logical) route. Their initial response:

A screenshot of a computer program

Description automatically generated

So, let’s follow their lead…

Created the form and understood the instructions enough to correctly customise and fix the **Copilot** response. For example (and this is great for the **README**), it asked me to add a new forms.py file. I already had one, so I could just add the ReviewForm to that, though I did have to know that the import Review needed to be added at the top. Also, the fields it chose were not in the right order or including all of the fields I wanted. Furthermore, it included a “content” field which doesn’t exist in the model. Fixed all of that and now onto the Views.

A screenshot of a computer screen

Description automatically generated

So that made sense to an extent. Again, I customised the imports, but the way that it’s handling code is different to other views in the file. They’re using HTTPResponseRedirect and a reverse instruction, whereas this is using regular render and redirect things. I’m sure there are benefits and pitfalls to each, and I’m not sure how to code this otherwise, so I’ll accept the suggestion for now and see how it works.

I did ask for further clarification on the comment they made so that I could fully understand it, and it gave a detailed explanation of what to change “review\_list” to and where to find the correct name. I need to get that from my urls.py file and it’s effectively the name of the page I want the user to be sent to on successful submission of their review. I think it makes sense to send them to the index page, so I’ll find the name for that and change it.

The name was “home” and I got **Copilot** to confirm that that’s what I should be entering. Very reassuring.

Now at this stage, I still don’t think I’ve done anything which might break my site. I’m running my server between each stage and making sure I save, git add, commit and push with good comments at every stage.

I tried to run a server and got an error:

“NameError: name 'login\_required' is not defined”

I asked **Copilot** to explain it and it highlighted that I hadn’t done the import at the top of the page. Again, *legendary* attention to detail (for anyone reading this who isn’t me, the point I’m making here is that I feel my attention to detail is generally good, and that I’m observant, so I’m sarcastically mocking myself). Corrected that and… my server loads.

*\*\*Phew\*\**

The next step is to create what looks like a super basic html page for the form to sit in. Here’s the **Copilot** instruction:

A screenshot of a computer program

Description automatically generated

I don’t understand what the {{ form.as\_p }} is referring to. Maybe it’s some Django code to input it as a paragraph element. That’s my guess. I’ll ask **Copilot** to explain. Also need to **pay attention** to whether crispy forms are needed/used and whether Summernote is included in the review text.

What do you know, I was right:

A screenshot of a computer program

Description automatically generated

Damn. I didn’t save, and do the gits after the previous stage. I’ve now also done the urls stage too, so I have 3 stages together. Not good. Here’s the urls stage:

A screenshot of a computer

Description automatically generated

Interestingly, **Copilot** has suggested doing the JavaScript and review approval after all of this has been done (presumably to not flood me with instructions):

A screenshot of a computer

Description automatically generated

So, now I need to see if the form, view and html have worked. Big moment…

It worked. It’s ugly, but it worked. I quickly saved and gitted everything. Here’s the initial view:

A screenshot of a computer

Description automatically generated

Trouble is, when I tried to enter a review it gave me an error.

The brief error is included in the below prompt to **Copilot** where I asked it to explain and help me resolve it:

A black screen with white text

Description automatically generated

**Copilot** did a great job of explaining the issue and guiding me on the fix.

A screenshot of a computer

Description automatically generated

It gave me a couple of options on how to fix it, but in essence, the problem was caused by the model having a NOT NULL in the status field, though I actually can’t see that in the models file. That may of course hinder the success of this fix. Bear that in mind if this still doesn’t work. I then added ‘status’ into the form to have it as a field for completion (**NB.** I need to add this to the wireframes), and a default value of 0 so that it goes through as Draft unless changed.

The fix worked, the review was submitted, and I have updated the wireframes. I set the status to Published on the review and then went into the admin panel to approve it to get it added to the index page. Then I saw 2 things: the status was “Draft” and there wasn’t a slug, which means the site would error out because you couldn’t get the review to load using the correct URL.

The solution I see is that the status correctly needs to default to Draft without the user seeing that (and it was interesting that the system changed it to Draft to keep with the rules of needing to be approved prior to publishing), and the field being hidden. The slug also needs to be in the form fields and again be hidden from user view. Not sure whether there needs to be a default value in the views.py file. I’ve asked **Copilot** if I’m correct in my understanding and approach and it’s said:

A screenshot of a computer

Description automatically generated

It then walked me through the steps, including adding “slugify” to the views.py. Fingers crossed it works.

It initially didn’t work. We included the fields and hid them from view using widgets, but then I got error messages on screen when submitting the review saying that the required slug and status fields were missing. I consulted **Copilot** again and removed the fields from the form (including the new widgets) and then it worked (as we’d already set a default field in the views for both – including the slugify).

I submitted a new review whilst logged in as “review\_bot” and it correctly gave me the confirmation message. I logged out and then back in as a superuser and viewed the review in the admin portal. Joy of joys, it was there, with a slug, and in draft waiting to be changed to published and approved.

And after a quick bit of gitting I then noticed that the box art isn’t coming through. Nice to see my old “missingboxart” image again! Need to fix it though.

A screenshot of a video game

Description automatically generated

A cartoon of a person with a question mark

Description automatically generated

I asked it how to solve this:

A black screen with white text

Description automatically generated

And it gave me an explanation which I don’t full understand and code which doesn’t mean much to me. As MVP is nearly here, and as a worst case scenario I can live with images not coming through, I’ll take a text copy of the files it’s changing as a backup.

It’s in the project folder and is called “Image Fix Code Save”.

I’ve applied the changes it suggested which were:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

NB – no changes here.

A screenshot of a computer program

Description automatically generated

NB. Only addition is the “request.FILES” bit. I don’t understand what that’s doing and I should probably ask it to clarify.

A screenshot of a computer program

Description automatically generated

NB. Again, only a small change – adding the enctype to the html. Again, don’t understand and should ask for clarification. Damn it, I will.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

I mean, it makes sense and it doesn’t. I get that that’s what the code is doing and that it should work, but I’ve not seen that code before and don’t think I needed it anywhere else in the project to make the Cloudinary file uploads work. We’ll see if the changes have been successful.

I then deleted the old review and submitted it again. Approved and published it. As a sneak peek, I checked Cloudinary and the image is there. So the code worked to get it sent up to storage. I viewed the index page and it’s there in all it’s glory.

A screenshot of a video game

Description automatically generated

A person wearing a hat

Description automatically generated

Now just need to link up the page to the nav bar and we should be golden for the morning’s work. I started off coding it myself, but then I got lazy (or worried I’d not get it right) and leaned on **Copilot** again. Good thing that I did too, as I had wanted to put the link at the end (after the “if” loop which checks for being logged in) as that matched my wireframes, and I even told **Copilot** that that was where I wanted it (even specifying the row and leaving a gap there). **Copilot** actually guided me to put it into the loop so that non-logged in users wouldn’t be able to see if. That saves me coding any “You’re not logged in” messages, so that’s cool. Totally makes sense, and it explained the rationale too.

Still very much need to sort the appearance of the submit review page so that it matches the others and utilise **Crispy Forms** and **Summernote.**

Going back to the original steps which **Copilot** gave me this morning, I need to add the JS code for the edit and delete buttons. That can be this afternoon’s job.

This afternoon:

1. Fix how it looks.
2. Integrate Crispy Forms and Summernote
3. Code Edit and Delete buttons
4. Update project board
5. Have a beer.

I’ve put 1 and 2 together and asked **Copilot** to help. Again, I could probably do this myself, but it’s quicker and easier to use **Copilot** and this is AI-Augmented Software Development, so…

A screenshot of a computer

Description automatically generated

**Copilot** asked me for a reference page and I gave it the about.html code as that has a form on it.

A screen shot of a computer

Description automatically generated

It correctly used Bootstrap for the styling. Let’s see what the code does…

A screen shot of a computer screen

Description automatically generated

So, it’s already brought in crispy forms (good job I mentioned I wanted those!) and used Bootstrap. The page before and after:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

That’s really good. I could live with that (except it doesn’t exactly match my wireframes).

Next up Summernote.

It checked that I have it installed and settings/urls configured correctly (which I did from previous Summernote usage), and then it just got me to add an import line and a widget to the forms code.

A screenshot of a computer program

Description automatically generated

Confirmed that now the text pane has Summernote. Time to do the gits and redeploy on Heroku.

Coding the Edit and Delete buttons is completely non-trivial. The Edit button needs to take the user to the submit\_review page so that they can edit everything about it. Just writing the 335-word prompt for this has been crazy. Here it is:

*My next task is one of the first ones I mentioned this morning which you suggested we should tackle later... Adding the ability for users to edit and delete their own reviews. Only the creator of the review should be able to edit or delete it (with exception of the admin superusers via the admin portal), and any edits need to go to the administrator for approval again. I've done this within this project before because on the review\_detail.html page users can make comments on reviews and they can then edit or delete their own comments (once they're approved). I can provide you with the code from that html page for you to use as a sort of reference if that would help. Similarly, there was some JavaScript code which helped the edit and delete buttons to work (in comments.js) and I can provide you with that code too. How this requirement will differ though is that there's no equivalent of the comment form within the review\_detail page (where the edit and delete buttons would need to appear to only the creator of the review) to handle the edit function. Upon pressing edit, the user would have to be taken back to the submit\_review page to allow them to make any changes that they would like to see, but the unseen slug field should not be amended. The user should then get a feedback message in the same way as they did when they submitted the review in the first place (i.e. being returned to home and getting a message at the top thanking them for their submission and confirming it's pending approval). The delete button should open a similar modal to the one the delete comment button opened, but saying something like "are you sure you want to delete your review". This all seems like a lot to ask. Also a lot for you to help me to code. Please could you break down the solution into steps and walk me through each one?*

This is going very slowly, but mainly because I’m not just accepting **Copilot** code, I’m checking through everything with the exception of the JavaScript, which whilst I can just about read and makes some sense to me, I really don’t understand it.

It’s taken me a super long time to just get the review\_html code to where I think it needs to be. I also need to **remember** to check how the buttons look and feel. The Edit review button is a <a> rather than a <button>, and is the only button that is, so that needs looking at. Also the buttons for editing and deleting the review are in a different order to the comments ones. Work out which way round is better.

The next steps went a lot smoother, probably 45 mins. It created this fantastic bit of code to change what text you get on the submit review page depending on whether you’re editing your review:

<h2>{% if form.instance.pk %}Edit Review{% else %}Submit a Review{% endif %}</h2>

Love it. So simple, yet makes the user experience much better.

I tested it locally and the edit function worked perfectly. The delete function didn’t though. I got a 404 error and gave the problem to **Copilot** to sort:

A screenshot of a computer program

Description automatically generated

It looks to be quite a lengthy fix though, which includes some more JavaScript.

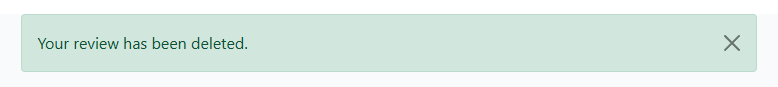
A screenshot of a computer

Description automatically generated

There’s something screwy with the javascript. It’s like we have 2 sets of code looking at the delete comment button. After all of this is done I need to check the functionality of the comments buttons.

There was a moment when I thought that it might be that I hadn’t run collect static, but now I have and it’s still not working. I’ve been around the issue with **Copilot** a few times, but don’t seem to be getting very far. Managed to get to the point where the 404 error doesn’t appear any longer (after I prompted **Copilot** to get me to consolidate the JS files), but the delete button in the modal just stays pressed and doesn’t do anything. We’re now debugging in the console. I hated this bit of JS, and of course it’s JS that’s tripping me up on this project. I should have been finished by MVP about 2 hours ago.

After going round and round, eventually **Copilot** tweaked the code enough that it worked.



Don’t think I’ve ever been happier to see a message like that!

Now to deploy to Heroku and see if it works there…

It didn’t.

404 error.

Gave the problem to Copilot, it suggested adding the django-heroku app and putting in some reconfiguration on the settings.py and wsgi.py files. Sure enough it worked.

The buttons don’t look similar to each other, and they definitely need to be the same way around:

A screenshot of a computer

Description automatically generated

**Check the button btn classes in the html and make sure that there’s nothing that will get broken if they change.**

The Edit also no longer works. AGAIN WITH THE JAVASCRIPT!!!

MONDAY. THAT’S FOR MONDAY.

Comment Delete works though.

All fixed. HOURS later. Here’s what I wrote to Richard:

“I was going through console logging the issue and I tried closing the modal instead of hitting the delete button and I had to hit the button twice. There seemed to be like a ghost/shadow modal closing the first time and it occurred to me that as there are two things generating modals on the page both of which were being triggered via JS, maybe they were firing simultaneously. I suggested that to Copilot and it saw the error and fixed it.”

**Day 9 (Monday 03/02/2025):**

Today starts with design and getting the final 2 Must-Have’s ticked off.

I’m looking up how to get a vignette style fade into the background and found this on Stack Overflow: <https://stackoverflow.com/questions/22216627/how-to-create-a-vignette-with-background-gradients>

**REMEMBER** to credit Jonathan Sampson if you use his radial gradient code.

This one is also nice and seems to have more controls:

<https://stackoverflow.com/questions/72564545/how-to-add-a-vignette-effect-to-an-image-using-only-an-img-tag-and-a-css-class>

**REMEMBER** to credit <https://stackoverflow.com/users/10867454/a-haworth>

I thought carbon fibre would be an interesting background image to apply the vignette to, though I struggled to find free images via my normal sites. I then used Leonardo.ai to generate one and also got one from doing a Google search. Which ever (if either) of those I choose will also need to be **CREDITED** in the **README**.

Leonardo.ai:

<https://leonardo.ai/>

Other image:

<https://www.freepik.com/free-vector/realistic-carbon-fiber-texture-3d-background_17820231.htm#fromView=keyword&page=1&position=2&uuid=4cea4853-b777-4f7f-9552-75637b8d8eca&new_detail=true&query=Carbon+Fibre>

<https://www.freepik.com/author/starline>

The vignette and background image didn’t work. I didn’t like the style in context and would prefer just a solid background.

I’m going to make 2 different sets of CSS variables, one for a darker mode and one for a lighter mode. Then I can switch them on and off by commenting in and out one set of css variables.

I scrapped that idea. I found a set of colours that I thought work well together using **coolors**. I set everything up and I like how it looks. I’ll check with Adri as she’s far more plugged into aesthetics than I am. I then sorted a logo using **Copilot** and customised it to fit in with the site. I also used the logo to generate a Favicon. All looks neat now.

I then recoded the index page so that clicking on the images links through to the detailed review page. Just a much better UX than clicking on the title (though that still works too).

Next up is to make the content match what’s on the wireframes after I’ve made sure it ticks the accessibility standards.

**Bug:**

The alt text on the index page just said the game title, and the review detail image didn’t say anything. I fixed these by adding “box art” to the game title on the index page, and correcting the code to review.game\_title (from review.title) and appending “box art” on the review detail page.

**Bug:**

The text blocks on the about me page look awful. The blocks themselves have a white background and the text is black. They should match the rest of the site, but seem to be overwritten by element.style CSS classes for p and h3 elements, which is likely to have come from Summernote. I have asked **Copilot** to help me fix it. It didn’t manage to sort it as it was right in saying it was a specificity issue, but I couldn’t get more specific than the inline styling that Summernote was forcing. I called up Spencer to get his opinion and with his guidance added “!important” tags to the specific css classes. **REMEMBER** to **CREDIT** Spencer for this help in the **README**.

**Bug:**

The Favicon is no longer working locally. The Favicons are only showing on index.html.

Asked Copilot who reminded me that static links should be using {% static… %} tags.

I put together a table of all of the colours I had interacting with each other and some failed their contrast checks, so I tweaked them a little to make them all pass. The colours all still work together, so I’m happy enough. That should be all of the accessibility tests now cleared.

Ok, now to make it look more like my wireframes.

Which reminds me – I need to create the filters and search bar. Damn. I’m not looking forward to that.

Good start. Added welcome message. Used Bootstrap to centre the text.

Next… The search function. Argghhh. Please be easy!

It wasn’t. Started off with Copilot and it was really difficult. Couldn’t get the actual form to appear on screen. Decided to try ChatGPT instead. Got so far and then ran out of credit. It got me to where I needed to be though and even added a link to direct users to submit their own review.

**Day 10 (Tuesday 04/02/2025):**

Today I need to sort the icons and filters. If I get chance I’ll do more, but that sounds like a lot for one day. Could be super quick if we manage to get **Copilot** to work to the best of its abilities.

I’ve decided what I need to use in order to display the icons (emojis for the stars, Font Awesome for the platforms and couch/online, and special font for the PEGI numbers) and now I need to start coding. I’m approaching **Copilot** in a different way today. Tell it what I want to do, but ask it how best to go about it and what information it needs from me to be able to work best. Here’s the prompt:

*“Today I have some fairly major changes to make to my site. I need to add icons/text onto the index and review\_detail htmls pages to indicate certain aspects which were selected during the review (such as which platform the game is available on) and I also need to give the user the ability to filter the game cards on the index page by those same aspects. Which part of this do you think I need to tackle first and what do you need to know in order to best support me in doing this?”*

**Copilot** gave me a very detailed and methodical approach to completing the tasks which starts like this:

A screenshot of a computer program

Description automatically generated

I’d really like it to hold my hand through the process though, so my next response to it is:

*“Thanks. I'd like to work with you on this step by step to make sure that my site works and doesn't get broken along the way. I'd therefore like you to provide me with instructions and requests for information that you need for each step, and to not proceed to the next steps until I have confirmed that I'm ready to. So, if we start with the beginning of your last response and the information you need, the aspects I'd like to display are the star rating (which can appear as it does in the models.py file - i.e. 1 star should be displayed as 1⭐), the co-op mode (which can be displayed using font awesome icons with <i class="fa-solid fa-couch"></i> for couch and <i class="fa-solid fa-globe"></i> for online), the PEGI rating (which can be displayed as "PEGI: " and then the number which was entered into the review form... as a further note, I plan to assign the number a different font later, so it would be great to give it a class or ID which I can use in CSS to adjust later), and which platform the game is available on (using font awesome again for the icons, being Xbox:*

*<i class="fa-brands fa-xbox"></i>; Playstation: <i class="fa-brands fa-playstation"></i>; Nintendo (there isn't one, so go with a big "N"): <i class="fa-solid fa-n"></i>; and PC: <i class="fa-solid fa-desktop"></i>).*

*I think that covers the information you said you'd need. Can we come to the filtering after we've sorted the icons?*

*In the step by step plan, we can skip step one as the Review model does have the necessary fields. Could you therefore start by walking me through the code to add the icons to the index page first, and then when I'm ready I'll let you know so that we can move onto the next step.”*

It went on from here to help me with the code, to guide me when I needed to correct my Font Awesome script, and to help me align the icons as I’d prefer for them to be displayed.

It’s also taking it slowly, step by step, and only moving on to the next step when I have confirmed that I’m ready to do so.

On the next step, rather than moving to add the icons to the review\_detail page it gave me the instructions to add the filters. We updated the forms, views and index.html files and it worked perfectly straight away. There are a couple of issues around the layout on smaller devices (gaps between fields when they’re stacked vertically) which I’ll note now as bugs to be fixed later:

**Bug:**

Fix appearance of filters on smaller devices is too close when they stack vertically and button positions also need to be addressed.

I changed the colour of the buttons to match the other buttons on the site.

And onto the next step…

We’ve now added the icons and repositioned them to beneath the banner (they were originally placed below the main review text). All looks good though there’s still an issue with a gap on smaller devices presumably caused by the images not loading on smaller screens. Added that as a bug to be fixed from previously noted bugs which are now displayed on the project board.

I thought that would be the end of the steps, so I asked **Copilot** to summarise what was left and it gave me these:

A screenshot of a computer program

Description automatically generated

It seems sensible to follow those steps through, especially as I have a **bug** around the appearance of the filter fields and buttons on smaller devices. May as well fix that now.

I gave it this prompt:

*I have tested those and the only issue I can see is that on smaller devices (767px width and below) the appearance of the filter fields and buttons isn't ideal. The search text field, the platforms and pegi ratings drop downs are too wide (nearly full width). Also, the top of the search button is too close to the bottom of the search text field. Id like to fix these using bootstrap if possible, but also not affect the appearance of the page on screens wider than 767px.*

It gave me this response:

A screenshot of a computer program

Description automatically generated

We went through a few tweaks, but eventually got it looking like it needs to.

The next step (after gitting and deploying) is to check the review detail page. Here is the prompt I gave **Copilot** so that it can help to address the other bug which is marked for fixing today:

*“The next issue to fix is that on the review\_detail page there is a large gap beneath the masthead section when the screen is 767px wide or below. When the screen is 767px wide and below the featured\_image also disappears, and I just have the masthead-text, review-title, review-subtitle and review-by lines at the top. I can think of two ways to fix this: first we could apply code that removes the section containing the featured\_image at screen sizes of 767px wide and below, or fix the existing code so that the image appears on the line below. If you can think of a third alternative that would be great. I'm leaning towards option 1. If you need any further information, such as that which I'd be able to get from the inspect screen in the browser, please let me know.”*

**Copilot** started by giving me the code which already existed. It turned out the code to hide the images was already on the page. What we now need to do is to adjust the CSS to remove any unnecessary padding or margins that might be causing the gap.

We adjusted the CSS and it got rid of the gap, but also caused an issue on screens larger than 767px wide – they were now getting full sized images which took over most of the page. The previous version was better. I have explained that and **Copilot** produced tweaked CSS code to attempt to fix it. That didn’t work as the images then disappeared completely and the gaps returned. It tried again to fix it unsuccessfully, so I rolled back the code to an earlier version (I copied and pasted a “safe” version of the code into a notepad just incase). I then suggested that media queries might be the best bet.

**Copilot** tried that way of doing things and it didn’t work. I then gave the problem to ChatGPT and it too failed (though I think both of them actually suggested the solution at one point). I then jumped on a Coding Coach call with Roo and he was able to see that it was the height: 33vh which was causing the gap. Once that was removed (along with the media queries) everything worked fine. **REMEMBER** to **CREDIT ROO** for his help in sorting this.

The next stage of my work for today is to create the “request profile deletion”

I got all of that done without too much stress. Did some tweaking and finished the main section of my coding. Everything from here is just correction and adjustments.

**Day 11 (Wednesday 05/02/25):**

Started today making sure my site matched my wireframes.

Got as far as the logout page when I realised that the “Submit a review” link in the navbar wasn’t going darker when the user is on that page. Thought is was a URLs thing… added a line into the URL… broke my site. Deleted the line and my site remained broken. Very scary. Closed the connection, reopened it and it worked. \*Phew\*.

**Copilot** suggested a tweak which I’m about to apply, but I realised that I’ve been using the Diffchecker website (<https://www.diffchecker.com/>) to compare suggested code to my own. It’s not always helpful as **Copilot** regularly uses double indentation, but for the most part it’s really helped to quickly confirm that the code **Copilot** has suggested is an exact match for the existing code. I must **REMEMBER** to **CREDIT** Diffchecker and note them as a used tool in my **README**.

It wasn’t immediately straightforward, but we got there.

I’ve now completed the matching to wireframes and also updated my About Me page to crop the image and update the text. I kept most of the AI generated text as it works really well.

Next… On to testing. That needs a new diary document, so I will start one now, imaginatively called “Testing.doc”.

Starting with the “Tidying Up” video, which handily starts with checking HTML.

First tested index.html. Only “info” alerts came up, which can be safely ignored.

Testing the detailed\_review pages though was a different matter. On the ones which **Copilot** generated, there were about 25 errors due to the rich-text nature of what was being copied and pasted into the Summernote field (it was trying to do things like stipulating fonts). I now have to “wash” every review to make it plain text.

Need to **REMEMBER** to note this in the **README** regarding the use of **AI**.

Been through an array of tests today and c.3.30pm am working on Lighthouse tests. Suggested conversion of image files to WebP format. Used <https://www.freeconvert.com/webp-converter> for this. **REMEMBER** to mention it in the **README**.

Lighthouse tests were extensive with a few proud moments as I achieved some 100% scores. Overall, pretty good:

A table with text and numbers

Description automatically generated

I then went on to the Python Linter testing and fixes. That was nerve wracking and at one point while I was testing that the create review function still worked, I thought I’d broken the site as I produced a 404 error. It was caused by me not marking a review as published before approving it. Now passed all Linter tests and my workspace is showing free of problems.

Tomorrow starts with docstrings.

**Day 12 (Thursday 06/02/25):**

I was talking to my daughter in the car on the way to school today about getting her to do a review of Stardew Valley, and she said that she would do it on her iPad, which I had mixed feelings about. Firstly, it’s good because I get to see it being tested on an Apple tablet device, so checking the device size as well as the Safari browser, though on the other hand the idea of her being able to attach a picture made it seem less likely to be successful. I thought back to a chat I had with Emma last week about not making every review form field mandatory. I decided at that point to keep them as mandatory as the site needs all of the data to be able to fill in the appropriate sections, though you can’t really expect users to go and research what PEGI rating a game is, or what platforms it’s on, and the same is true of copyright free images.

I will therefore create a user story to say that as a user I don’t expect to have to source too much information in order to provide a review, and then change the fields to be non-mandatory. The fields in question are: Co-op mode (x2), PEGI rating, Platform availability (x4) and Image.

I have used **Copilot** to check how to do this and make sure that I’m not making any glaring mistakes, especially around the database.

I will take a copy of my models file before changing it, though once I’ve made migrations to the database it will likely make things more complicated to undo.

The Co-op mode and Platform availability are already non-mandatory due to them being BooleanFields, so it’s just the PEGI rating and image fields I need to adjust.

All done – no issues. On to the rest of tidying and testing.

Sorted Meta section on base.html.

Did all of the docstring code and discovered I’d missed a load of PEP8 compliance from the review section. Fixed all of that with the exception of an erroneous “review” variable which isn’t causing an issue but doesn’t relate to anything.

I had a session with Emma where she had a play with my site and discovered the bug where an unapproved review appears in the index and results in a 404 error. I need to fix that.

I also need to get images of all of the pages which are free of linter errors.

Emma suggested that I comment out the line which includes the “review” variable to see if it breaks anything.

Fixed the error caused by draft and unapproved reviews appearing in the index by tweaking the views.py and models.py code.

Taken screenshots of all clear linter pages.

Removed the erroneous local variable.

I produced a test report for Richard based on his site and he’s applied a number of amendments following my recommendations.

I started working on my ReadMe document (perhaps following Emma’s template a little too closely), and whilst writing about my index page filters I tried filtering for PlayStation games to ensure that the filter worked correctly on subsequent pages. It didn’t. The filter was dropped and the standard page 2 was displayed. Massive **bug**.

I wrote the following, very nervy, prompt to **Copilot**:

A black screen with white text

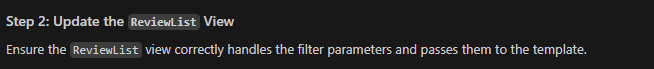
AI-generated content may be incorrect.

It’s following my request to the letter:

A screenshot of a computer

AI-generated content may be incorrect.

Step 2 was:



Having completed these using the **Copilot** suggested code, the **bug** has now been squashed.

Carried on working on Readme and got as far as completing wireframes.

Got Adri to do user testing by following these instructions:

1. Using a 13” laptop and the Edge browser, navigate to the site and browse around to make sure you’re familiar with it and comment on anything that doesn’t feel familiar.
2. Register as a user
3. Write a review on “That’s You!”
4. Comment on 2 other reviews. Edit one comment and delete the other.
5. Log out

As part of this we discovered that the text on the sign in, sign up and password reset forms was white text on a white background. I have since raised a **bug** and fixed it.

Jon and Izzy both also tested the site and this is their feedback:

“Excerpt for index page” on the review form confused them. Use language which would be more accessible. The text field is also far too big, which adds to the confusion – making them feel like they should be writing more.

When registering, Jon noticed that the username needs to be one word, and that isn’t explained. He had to work it out himself. This should be explained on the signup page.

Izzy discovered that if you change the font in the Summernote text section of the review it feeds through to the published review. This needs to be overwritten to force the site’s main font.

The comments don’t appear to be in any order (perhaps alphabetical?). It would be better to have them by oldest first, but that would need a field to be added to the model with the date & time in it. That’s too much to do at this stage, so I’ll add it as a “Could-Have” bug.

Richard also followed through the password reset process and found un-styled pages along with references to “example.com”. I need to investigate and fix this.

**Day 12 (Friday 07/02/25):**

Frustrating start to the day. The first half hour went well in that I fixed the **bugs** around the index excerpt. The other **bugs** have not been fixed though. The username issue requires adding a new model and customising 2-3 Python files. The Summernote issue requires JavaScript to fix. I’ve only got to this stage by lunchtime.

I’ve moved the user stories to their own document and given just one example of the format/layout on the readme.

I’ve also fixed the link to my deployed site (it had a double https:// in it).

Done more work to readme. Need to add amiresponsive image to cover all features.

**Day 13 (Sunday 09/02/25):**

An extra weekend day to make additions to the Readme. In short, most of the testing section was done. Added in contents pages to make navigation much easier on lengthy files.

Checked through the learning objectives (particularly the AI ones) with Adri to ensure all had been adequately met.

**Day 14 (Sunday 09/02/25):**

All being well, the final day of development. I have attached this diary document to the readme (though it will need to be reuploaded to ensure the final version is on there.

Emma mentioned this morning that we need to ensure that all commented out code has been removed from our workspaces, as well as any files which are unused (such as the test.py files). I’m pretty sure I’ve sorted my commented out code, but I’ll double check. As far as the test.py file is concerned (and any others) – I’m a little nervous about removing them. What if there’s a link to those files within say, the settings.py file, and that removing the file gives that connection nothing to connect to (even though it’s not doing anything) and it results in my site failing.

I will take make amendments, and thoroughly check the function of my site locally before pushing the changes.

So, today I need to:

* Finish the readme [done]
* Check through any files not used within my windows project directory [done]
* Read this dev diary to see if I’ve missed any credits/acknowledgements [done]
* Go through CI resources to check off everything [done]
* Future development point – add email notification of reviews and comments [done]
* Ensure commented out code is removed [done]
* Delete test.py and any other unused files [done]
* Update readme following Emma’s feedback (add screenshots to the Features section, delete the paragraph about automated testing, check spellings of “editing” [done]

A note on this diary as a whole – it’s been a useful tool to facilitate thinking out loud. I started using a Miro board for the same purpose and it was great to map ideas out and really get started, though it quickly became a version of this diary and kept almost as a tool to remind me where to start the next day. As a result, it got phased out and replaced by this diary (and hence why this diary only starts on day 4). As far as my workflow goes, I think this has worked well for me. Miro to ideate and plan the initial stages, and then a dev diary (though perhaps less detailed so that it doesn’t end up being a near 50-page document).

And just like that, I think I’m done. The last things to do are to re-add this document to the repo via my IDE, git add, commit and push one last time, check my git status is clean, redeploy on Heroku, give it a quick test, tick off the last Learning Objectives in the project board, re-read the objectives as a sanity check and then submit the project (I’ll hold fire on that last step for 24 hours just in case someone says anything that triggers me to do anything else!).